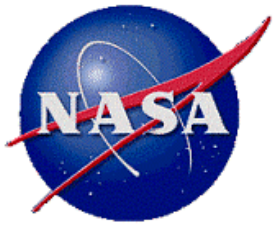


Software Engineering Research Infusion

Tom Pressburger, Research Infusion Lead (ARC)
Ben Di Vito (LaRC), Martin Feather (JPL),
Michael Hinchey (GSFC), Lawrence Markosian (ARC),
Tim Menzies (Portland State Univ., IV&V),
Luis Trevino (MSFC)

Outline

- Background
- Selected software engineering technologies

Background

- NASA Software Engineering Initiative
  - Led by the Office of the Chief Engineer
  - Improve software engineering to meet the challenges of NASA
  - Some of the areas of activity
    - Improving software development process
    - Training the workforce
    - Improving NASA guidelines, policies, procedures
    - and….

Infusing Software Engineering Research

- Goal: Transfer into practice
  - NASA-sponsored Software Engineering Research
  - Other new software engineering tools and technologies

- Approach
  - Present selected technologies to the NASA software development community, and
  - Encourage and support collaborations between the researchers and NASA software developers.

Collaborations

- Initiated by a software developer interested in one or more of the technologies.

- Purpose
  - **benefit the software development project**
  - validate the research
  - Not: further develop the research

- Funding available for—
  - training and consulting in the use of the technology
  - license fees in the case of commercial technologies
  - managing & applying the technology,
  - collecting & analyzing data
  - reporting results.

Funding for Collaborations

- Funding for several small projects available from OSMA via the Software Assurance Research Program (SARP).
  - Last year: 6 projects in the range $15K - $40K.
  - Customer as PI (in consultation with the technology provider) submits a collaboration proposal.
  - Proposal template and instructions on the Research Infusion website.
  - Due: June 28, 2004, 9 AM Pacific
  - Start: February 2005
- We will help facilitate unfunded collaborations.

Current Collaborations

- These collaborations applied technologies presented in September 2003 Research Infusion ViTS
- 6 projects selected out of 13 submitted proposals
  - DSN Antenna Controller (JPL)
  - Flight software projects (GSFC, MSFC)
  - Station software (ARC, JSC, USA)
- Technologies
  - Static code analyzers
  - Formal inspection technique
  - Defect classification technique for process improvement
- Several of these technologies are available again this year for collaborations in 05.

Selected Technologies

- Culled from
  - NASA-sponsored software engineering research
  - Leading edge commercial tools
  - Research in DoD, institutes and industry suggested by NASA projects.
- Reviewed by researchers at several centers experienced in tech transfer of software engineering research.

Selected Technologies (continued)

- Technology Selection Criteria
  - Has been successfully applied
  - Easily adopted
  - Focus on Software Assurance

Collaboration Roles

For Technology Provider and Software Development Team

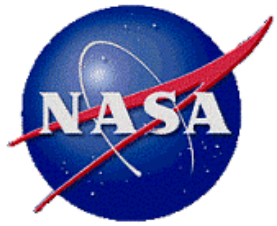
- Technology provider
  - During proposal preparation: help plan collaboration, including help select suitable application
  - 1 – 3 day training course at your site
  - Online tutorial and other user documentation
  - Customer support throughout collaboration

Collaboration Roles (continued)

- Development team
  - During proposal preparation:
    - Contact Research Infusion team, let us know you plan to submit proposal
    - Work with technology provider to plan collaboration and select suitable application
    - Write and submit the proposal
  - Take training course
  - Identify software artifacts to which tools will be applied
  - Apply the technology, sometimes in multiple iterations
  - Collect data & evaluate performance

Next Step

- If you're interested in a collaboration involving one of the selected technologies, follow the proposal process at http://ic.arc.nasa.gov/researchinfusion/
- We want to provide feedback on proposals before the due date.



Selected Technologies

Lawrence Markosian

Selected Software Engineering Research Technologies

- Software Cost Reduction
  - Toolset for formalizing, analyzing and verifying software requirements specifications
- SpecTRM
  - Toolset for formalizing, analyzing and verifying software requirements specifications
- Software Architecture Evaluation
  - Tools and methodology for verifying that source code implements the intended design
- Usability & Architecture*
  - Methodology for verifying consistency of architecture with usability requirements
- Orthogonal Defect Classification for NASA*
  - Process improvement methodology

* Re-offered from last year's list

Selected Software Engineering Research Technologies (continued)

- UML Tools
  - Design quality evaluator
  - Test Development Environment
- MATT
  - Verification and testing tools for Matlab models and Matlab-generated code
- FLUID
  - Static source code analysis technology for detecting race conditions & certifying absence of race conditions in Java code.
- CodeSurfer *
  - Reverse engineering/debugging toolset

* = Re-offered from last year's list

Technology Description Format

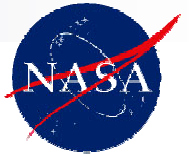
- What problem does it address
- What is it
- Features
- Benefits
- Successes
- Contexts for best use

Software Cost Reduction

PoC: Connie Heitmeyer, Naval Research Laboratory

- What problem does it address?
  - Incomplete, inconsistent specifications that cannot be rigorously analyzed or tested
  - Late, costly detection of requirements defects.
- What is it
  - Suite of tools for specifying and analyzing system and software requirements.
  - Specifications are based on state machines
  - The tools support the construction, validation, and formal analysis of requirements specifications in the SCR tabular notation.

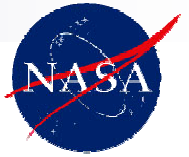
Software Cost Reduction (continued)



- Features
  - Specification editor for constructing the specification
  - Consistency checker for automatically detecting well-formedness errors (such as syntax and type errors, missing cases, and undesired non-determinism)
  - Simulator for validating that the specification satisfies the customer's intent
  - Various formal analysis tools for checking that the specification satisfies critical properties, such as security and safety properties.

- Benefits
  - Detection and removal of requirements errors early in the lifecycle.

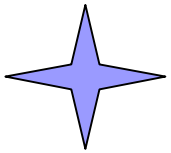
Software Cost Reduction (continued)



- Successes
  - JPL used SCR specification editor, consistency checker, and simulator to develop a specification of a complex software component of a Fault Protection Engine (FPE).
    - Constructing specification required only three weeks
    - Clarifying the underlying FPE requirements.
  - Lockheed Martin using SCR to specify and analyze applications, such as
    - autopilot logic,
    - flight navigation,
    - flight control and management,
    - airborne traffic and collision avoidance.

Software Cost Reduction (continued)

- Contexts for best use
  - Safety-critical, embedded systems and software
    - The additional up-front cost of developing high-quality requirements is justified by increased system and software reliability.
  - Users should have
    - solid information about the system (or component) requirements and
    - ready access to domain experts who can answer questions about requirements and aid in validating them.



SpecTRM

PoC: Grady Lee, Safeware Engineering
NASA Funding for underlying research

- What problem does it address?
  - Incomplete, inconsistent specifications that cannot be rigorously analyzed or tested
  - Late, costly detection of requirements defects.

- What is it
  - SpecTRM is a system development environment with particular emphasis on mission-critical and safety-critical systems.
  - SpecTRM facilitates construction of software requirements models that can be simulated and analyzed.
  - State machine-based specifications

SpecTRM

PoC: Grady Lee, Safeware Engineering

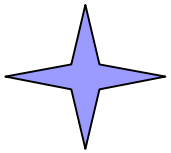
- Features
  - Emphasis on construction of software requirements models that can be easily read, reviewed, simulated, and analyzed
  - User-friendly editing environment for recording system-level requirements and design and black box software requirements models.
  - Hyperlinks in SpecTRM ensure traceability of safety information and design rationale from the system level through component development.
  - SpecTRM requirements modeling language enables model-based system design.
  - Static analysis & simulation to support defect detection at the requirements level.

SpecTRM (continued)

- Benefits
  - Helps engineers build desired properties into the system at the beginning and ensures that those properties are embodied in the design.
  - Find errors at the requirements level, where resolving errors is least costly and most effective.
  - Facilitates tracing requirements information, specifying design rationale, and updating safety information throughout life cycle.
  - Reduces time required to design new aircraft & spacecraft components by reusing previous requirements.

SpecTRM (continued)

- Successes
  - Derived from NASA & FAA work on TCAS II
  - Adopted and used by Japan Manned Space Systems Corporation
  - SpecTRM-based services provided to automotive, aerospace and medical devices industry by Safeware
    - Including an electric steering system for Delphi Automotive
- Contexts for best use
  - Software-intensive, mission-critical and safety-critical systems.
  - Software with complex decision-making algorithms, such as mode and state transition logic, benefit more than systems where complexity is in numerical calculations.



Software Architecture Evaluation

PoC: Mikael Lindvall, Fraunhofer Center – Maryland

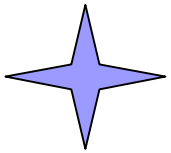
- What problem does it address?
  - Risk that implemented system's architecture does not match the intended architecture
  - Increased maintenance difficulty as deviations occur from intended architecture during maintenance
- What is it
  - Process & toolset that assure that the source code implementation of a software architecture matches the architecture.
- Features
  - Tailorable to project needs, architectural styles, design patterns, general guidelines and design rationale.

Software Architecture Evaluation (continued)

- Benefits
  - Quickly check that the source code conforms to the planned architecture.
  - Identify architectural violations & prevent architecture from degenerating during maintenance.
  - Assure that architecture remains flexible despite software evolution.
  - Make reviews more efficient by ensuring the architecture is accurate.

Software Architecture Evaluation (continued)

- Successes
  - Applied to several research projects and one commercial product.
- Contexts for best use
  - Can be applied to Java and C/C++ projects.
  - Object-oriented systems based on design patterns, architectural styles, etc. benefit the most.



Usability & Architecture

PoC: *Bonnie John, CMU*

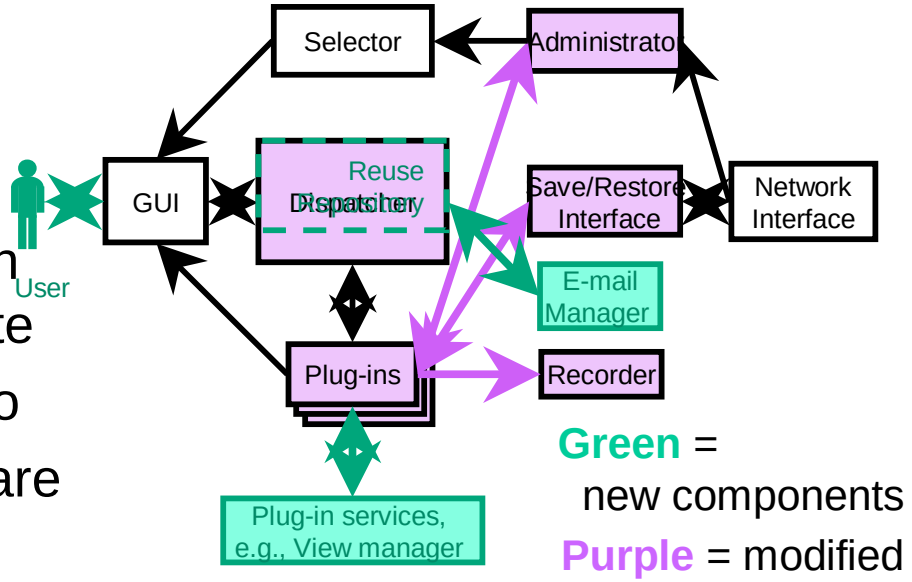
Funding: Code R, Engineering for Complex Systems and Communications, Information, and Computing Technology programs, High Dependability Computing

- What problem does it address?
  - Reduce risk that the *software architecture* of an *interactive system* has to be changed due to usability concerns.
  - "Yikes! You mean we CAN'T CANCEL COMMANDS??!!"

Oh no!

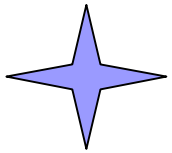
Usability & Architecture (continued)

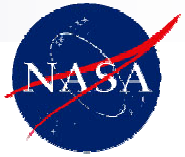
- What is it?
  Methodology with—
  - 27 usability scenarios:
    - e.g., cancellation, information reuse, observing system state
    - Benefits of including scenario
    - Responsibilities of the software to support the scenarios
  - Methods for evaluating applicability of the scenarios
  - Architecture patterns that support the scenarios



Usability & Architecture (continued)

- How to use it
  - At architecture design (or redesign) time:
    - Consider the usability scenarios
    - Decide which are important for the application
    - Ensure that the proposed software architecture fulfills responsibilities listed for those scenarios
- Benefits
  - Avoid architecture decisions that impair usability
- Successes
  - Modification of MERBoard's architecture, based on a usability analysis.





Orthogonal Defect Classification

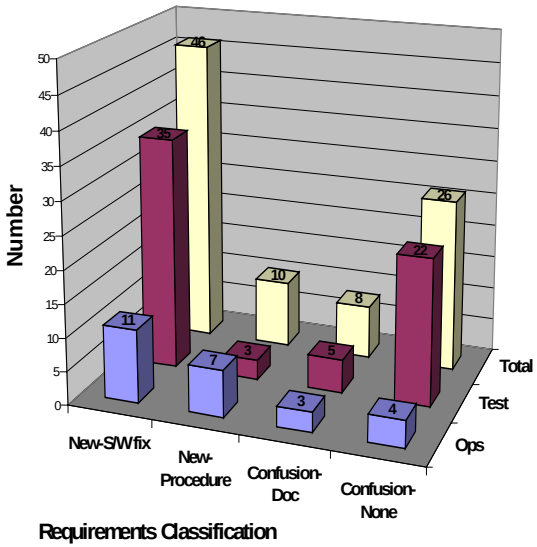
PoC: Robyn Lutz, JPL

Funding: Office of Safety and Mission Assurance, Software Assurance Research Program; and National Science Foundation

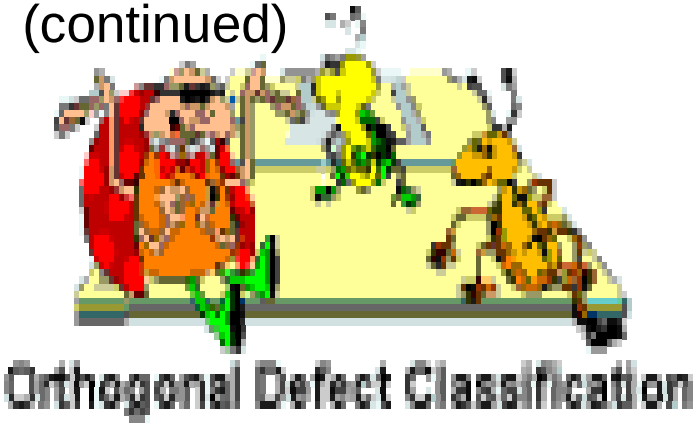
- What problems does it address?
  - Process improvement – learning lessons from defect logs
    - Currently: defect logs in many incompatible formats
    - With ODC: generalized schema for defect logs

Orthogonal Defect Classification (continued)

- What is it
  - Method for analyzing software bugs to determine patterns and improve software development process
  - First developed ~1990 by Ram Chillarege at IBM, now widely used in industry
  - When faults are first seen:
    - record “activity” and “triggering event”
  - When faults are fixed:
    - record “target” and “type” of fix



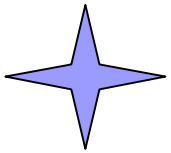
Orthogonal Defect Classification (continued)



- Features
  - Language, platform independent
  - Produces customizable Excel graphs
  - Much local expertise
  - Useful to single project or to organization
- Benefits
  - Provides quantitative basis for process improvement
    - Establishes a baseline for patterns of software defects
    - Much less expensive than root-cause analysis
  - Provides guidance in allocating funds for post-launch maintenance
  - Enables effective corporate memory

Orthogonal Defect Classification (continued)

- Successes
  - Analyzed ~800 testing problem reports from Mars Exploration Rover
    - Identified mechanisms by which requirements changes occur and are resolved during testing and operations.
  - Current Research Infusion collaboration with JPL Antenna Beam Waveguide Retrofit project
  - Adopted by companies such as IBM, Motorola, Telcordia, Cisco, and Nortel
- Contexts for best use
  - Teams that want to incorporate improved defect metrics into their development or maintenance process.
  - It “looks at the forest, not the trees” to identify defect patterns of concern.



UML Tools: Design Advisor & Test Development Environment

PoC: Jean Hartmann, Siemens Corporation

- What problems does it address
  - UML designs that do not follow standards
  - Difficulty in generating test cases
- What is it
  - Design Advisor: tool for assessing and measuring the quality of UML models according to a set of UML-specific guidelines.
  - Test Development Environment (TDE): tool for automatic black-box generation of conformance tests from UML diagrams including Statecharts, Sequence and Activity diagrams.
  - Both tools available as Rational Rose Add-ins.

UML Tools: Design Advisor & Test Development Environment (continued)

- Features
  - Design Advisor
    - Checks can be executed for individual UML diagrams or packages as well as an entire model
    - Violations have detailed explanations to guide inexperienced analysts and designers
    - Violation and metrics results can be exported in a variety of formats.
    - Supports customization of new design guidelines and metrics by users.
  - TDE
    - For unit and integration testing, the tool derives test cases that validate not only the individual components, but collections of components that interact.
    - Derivation of tests from Use cases and Activity diagrams provides support for system testing.
    - XML-based output format allows the test cases to represent executable code, executable test scripts or textual test procedures/test steps

UML Tools: Design Advisor & Test Development Environment (continued)

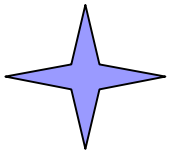
- Benefits
  - Design Advisor
    - Earliest possible detection of defects relating to architectural analysis and design.
    - Helps enforce quality standards
  - TDE
    - Supports definition of UML-based test specifications
    - Provides a notion of functional test adequacy as part of the generation process
    - Reduces test script maintenance by regenerating test cases when test specifications have been updated.

UML Tools: Design Advisor & Test Development Environment (continued)

- Successes
  - Design Advisor
    - Applied within Siemens in several application domains including industrial automation, hospital information systems and transportation control systems.
    - Analysis of UML models revealed numerous defects.
  - TDE
    - Applied within Siemens to several application domains including medical imaging, telecommunications, hospital information systems, and industrial automation.
    - Yielded reduction in testing cost and effort.

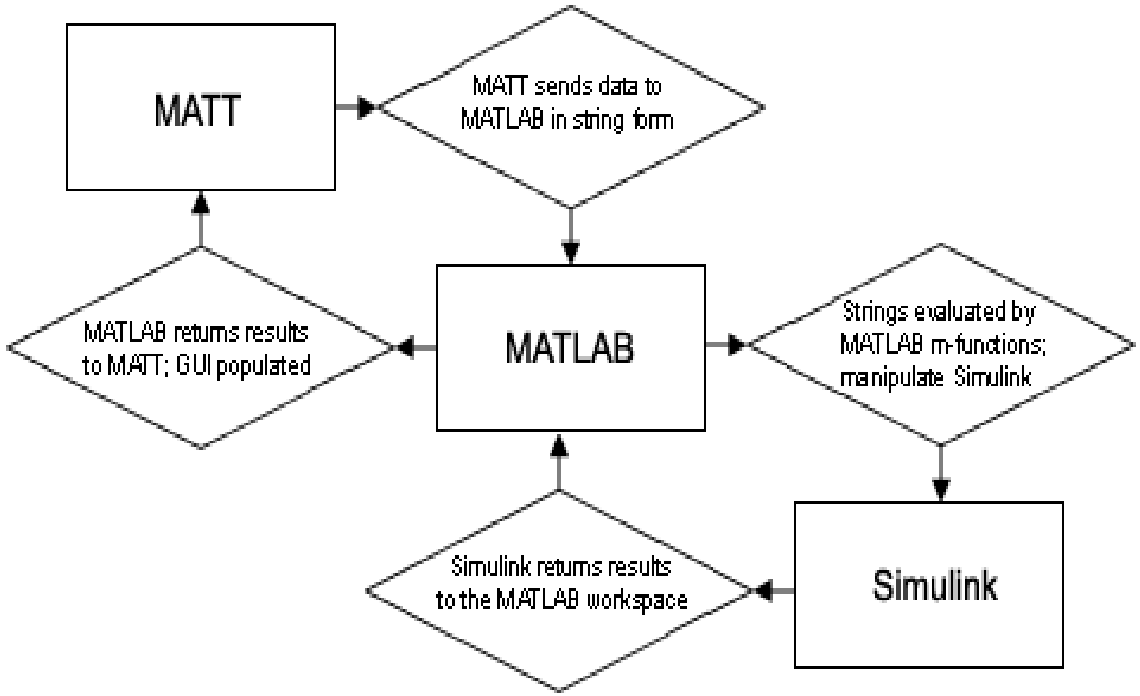
UML Tools: Design Advisor & Test Development Environment (continued)

- Context for best use
  - Design Advisor
    - Best used by requirements analysts, architects and designers working with UML-based models and visual modeling tools such as Rational Rose.
    - No additional effort required to apply the tool.
  - TDE
    - Best used by test designers or lead developers when defining unit, integration or system tests.
    - Must be willing to make the additional effort required to annotate the UML diagrams



MATT: Matlab Automated Test Tool

PoC: Joel Henry, University of Montana
Funding source: NASA OSMA/SARP



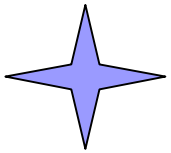
- What problem does it address
  - Automated generation of test cases for Matlab/Simulink models and generated code
- What is it: Tools for –
  - specifying test data for inports, test step duration and total test tim
  - defining defect criteria for outports,
  - executing a simulation, and
  - investigating the output values for each outport manually and through graphs.

MATT: Matlab Automated Test Tool (continued)

- Features
  - select the entire Simulink model, or any subsystem for testing;
  - specify the time step and test duration;
  - quickly generate the input values for each inport for each time step using built in functions, user specified functions, and graphical tools;
  - specify defect criteria for each outport or combination of outports;
  - quickly execute tests using simulation or test the automatically generated code;
  - detect defects based on outport defect criteria; and
  - configure and execute tests on a system or subsystem based on combinations of inputs.

MATT: Matlab Automated Test Tool (continued)

- Benefits
  - Quickly configure and execute tests on Matlab/Simulink models.
  - Re-use tests when the models change and source code is regenerated.
  - Detect and evaluate defects quickly.
- Successes
  - Used by SAIC on the STEREO (Solar-Terrestrial Relations Observatory) project (GSFC)
- Context for best use
  - Developers and testers familiar with Matlab/ Simulink
  - Development environment where model developers can use MATT consistently.
  - NASA IVV Facility evaluation report specifically recommends it for unit testing



FLUID: Race Condition Detection/Certification Tool

PoC: William Scherlis, CMU
Funded by NASA High Dependability Computing Program

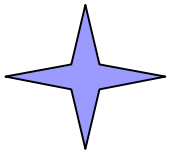
- What problems does it address
  - Detection of race conditions in Java applications
  - Certification of absence of race conditions
- What is it
  - Tool for analyzing Java source code to detect potential race conditions and in some cases, assure their absence
  - Integrated into Eclipse open source development environment
  - It is intended for use by developers actively evolving a Java code base
  - It can also provide retrospective assurances for existing code.

FLUID (continued)

- Features
  - Supports javadoc-style declarations of design intent as program annotations
  - Uses static analysis to assess consistency of the code and the models expressed using the program annotations
- Benefits
  - Static assurances for critical multi-threading properties that are difficult or impossible to assess using traditional testing, inspection, & runtime checking methods.
  - Reduces likelihood of introducing race conditions during maintenance.

FLUID (continued)

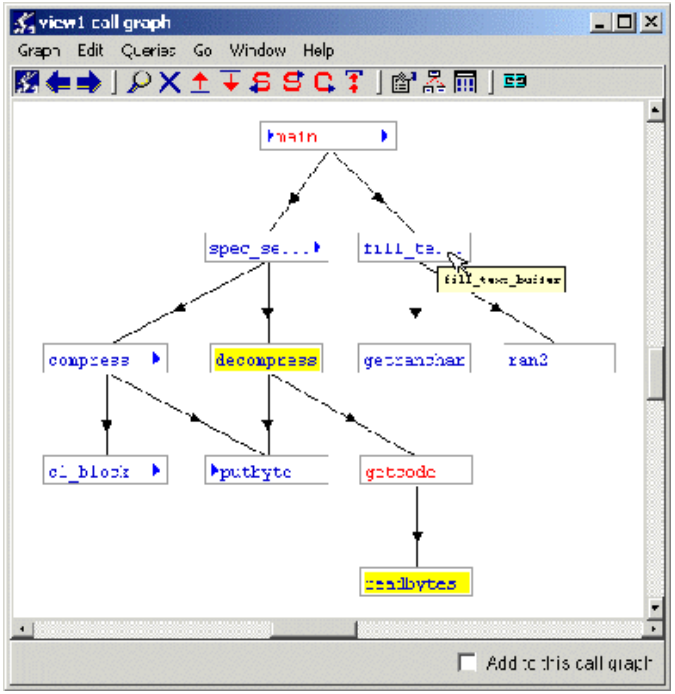
- Successes
  - Experimentally applied to wide variety of Java production systems and components, including NASA's CTAS
  - Found potential races (not false positives) in nearly all larger systems, including widely used library code.
- Context for best use
  - Works most effectively on Java systems that are decomposable into subsystems sized 50KLOC or less.
  - Focus is on lock-based concurrency
  - Most readily applied to new development
  - For new or old development, users must be willing to provide the required program annotations



CodeSurfer

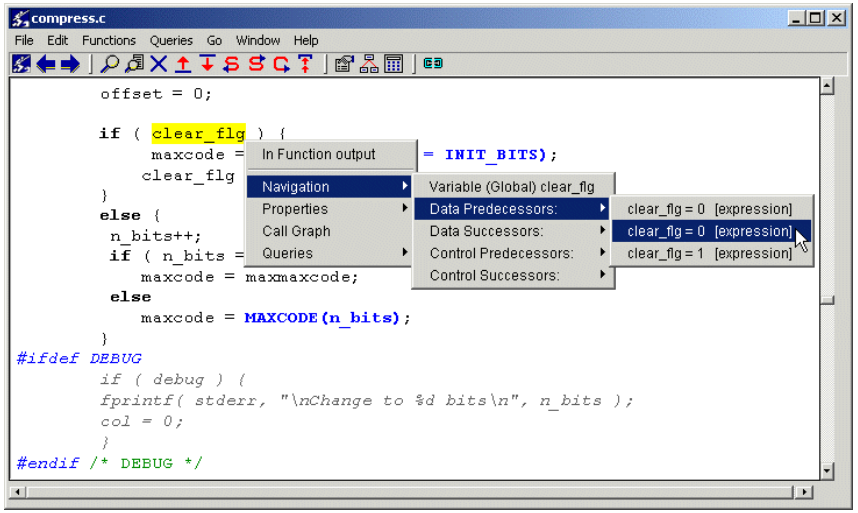
PoC: Mark Zarins, GrammaTech, Inc.

- What is it: C Source code analysis tool using—
  - Program Slicing
    - Highlights code relevant to understanding a particular issue
    - Does impact analysis
  - Pointer Analysis
    - Tracks loads and stores via pointers
    - Takes indirect function calls into account
  - Buffer overrun detection (with plug-in)
- What problem does it solve:
  More efficient—
  - Reverse engineering
  - Debugging
  - Safety/Security auditing
  - Documentation



CodeSurfer (continued)

- Successes
  - Mitre, MIT, Thales, Network Associates
  - Recently obtained by NASA MSFC
  - Collaboration at JSC, not yet evaluated
  - One user (at a large aerospace company) reports:



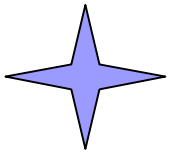
"Without CodeSurfer, [manual analysis of defect root cause] required about 2 to 5 days full time for one person. When using CodeSurfer, the same task has been reduced to 2 hours."

CodeSurfer (continued)

- Features
  - Generates interactive, graphical reports
    - Trace data flow backward and forward through code
    - Display what variables a pointer can point to
    - Highlight code that affects selected statement(s) and/or variable(s)
    - Call graph
    - Change impact analysis, etc.
  - API for customization and batch processing
  - Commercially supported product (CodeSurfer base product, not including buffer overrun detection plug-in)
  - Approx. $2000 for single-seat floating license with 1st yr maintenance contract (without API)

CodeSurfer (continued)

- Contexts for best use
  - Need compilable C source code; build application with CodeSurfer.
  - Best applied on applications of up to 100K – 500K LOC.



Next Step

- If you're interested in a collaboration involving a Research Infusion technology, check out the collaboration proposal process at

  http://ic.arc.nasa.gov/researchinfusion/

- We will help broker matches of technology and software developers.

Research Infusion
Software Engineering Technologies

More documentation AND the proposal template for collaborations are available at the Research Infusion website:
http://ic.arc.nasa.gov/researchinfusion

- Software Cost Reduction
- SpecTRM
- Software Architecture Evaluation
- Usability & Architecture
- Orthogonal Defect Classification for NASA
- UML Tools
- MATT
- FLUID
- CodeSurfer